Tigres: Template Interfaces for Agile Parallel Data-Intensive Science

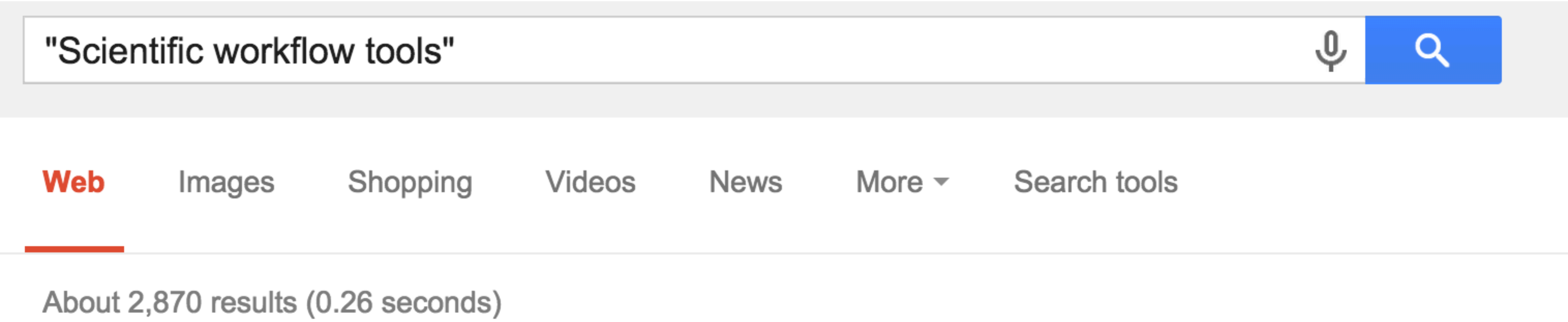
Lavanya Ramakrishnan

LRamakrishnan@lbl.gov

U.S. DEPARTMENT OF ENERGY | Office of Science

CRD computational research division

Google says there are a lot of workflow tools available ….

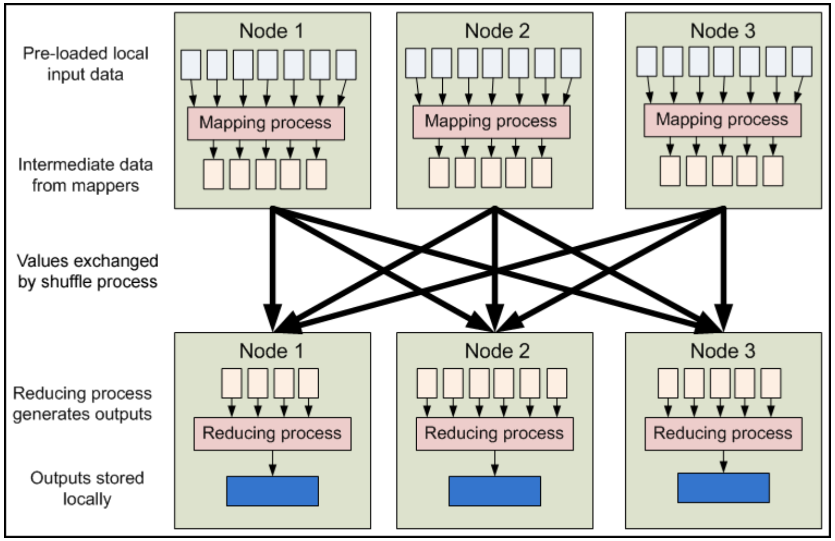


"Scientific workflow tools"

About 2,870 results (0.26 seconds)

2,870 results
237,000 results for “workflow tools”

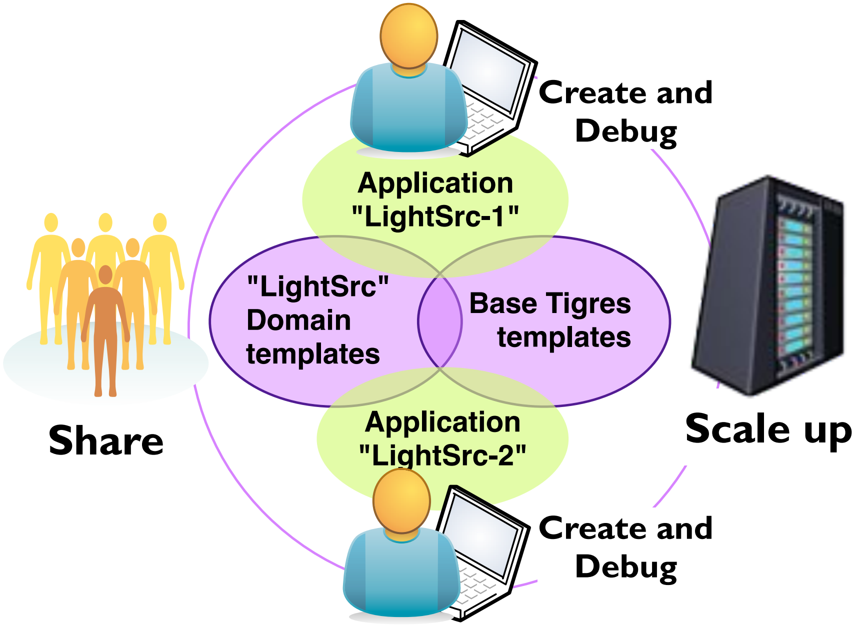
MapReduce/Hadoop



U.S. DEPARTMENT OF ENERGY | Office of Science

CRD computational research division

Tigres: Design *templates* for common scientific workflow patterns



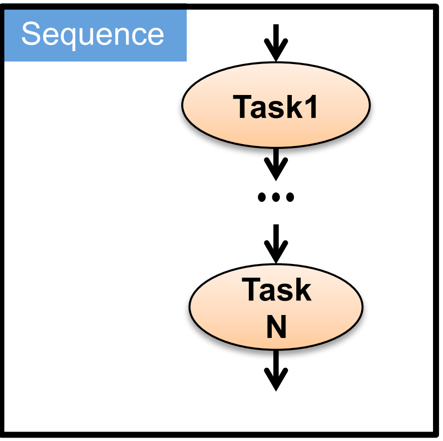
Workflow Library: Implement templates as a library in an existing language

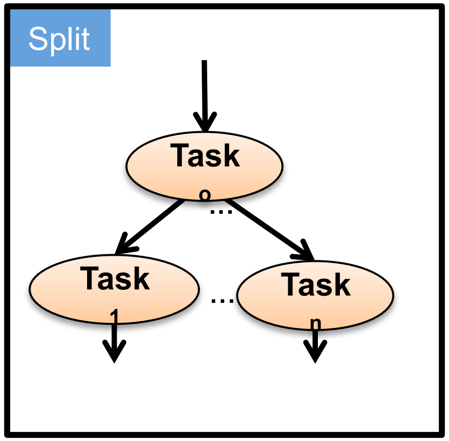
Basic Templates: Sequence, Parallel, Split, Merge

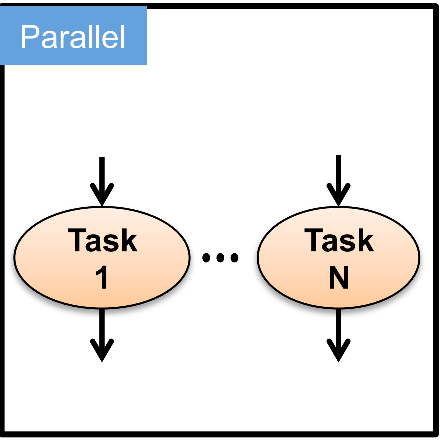
Key Aspects of Tigres

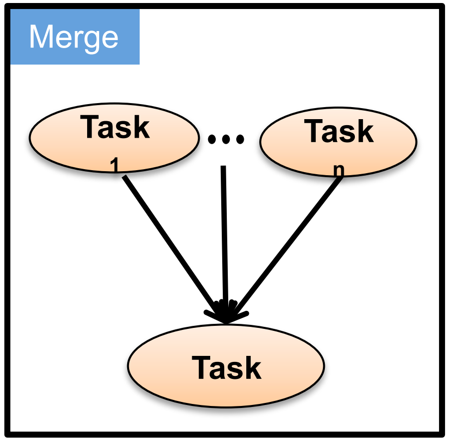
- Targeted for large-scale data-intensive workflows
  - Motivated by “MapReduce” model
- Library model embedded in existing languages such as Python and C
  - “Extend current scripting/programming tools”
  - API-based, embedded in code
- Light-weight execution framework
  - “As easy to run as an MPI program on an HPC resource”
  - No persistent services
- User-Centered Design Process
  - Get feedback from user continuously

Tigres Templates









Sequence (name, task_array, input_array)

Parallel (name, task_array, input_array)

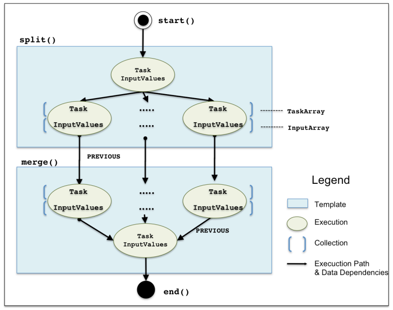
Split (name, split_task, split_input_values, task_array, task_array_in)

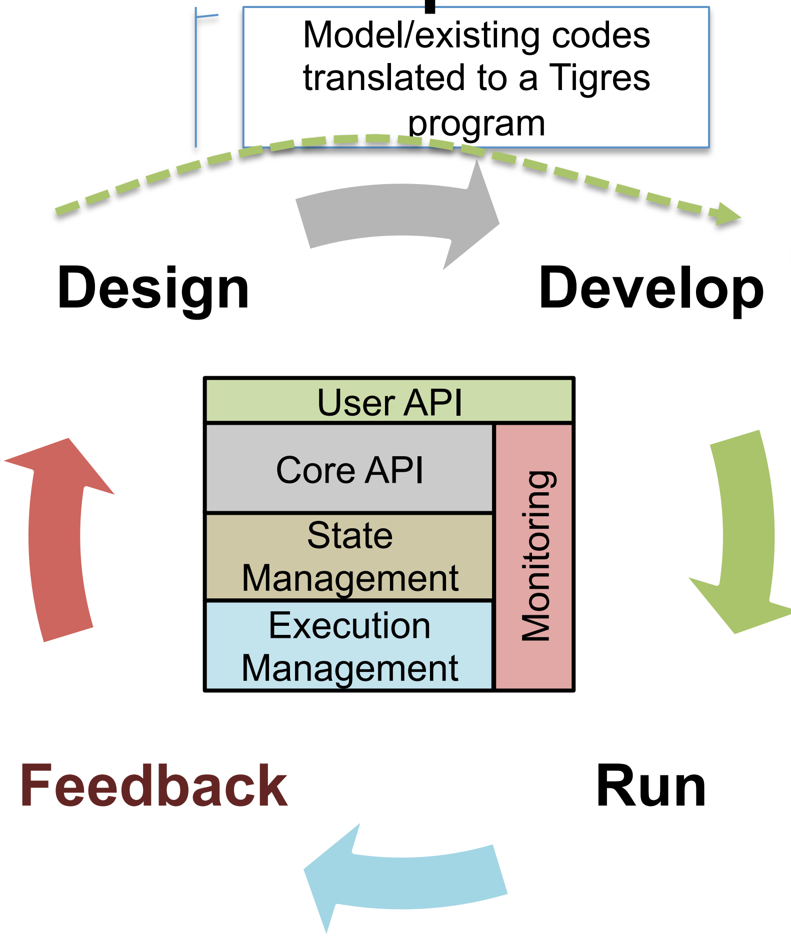
Merge (name, task_array, input_array, merge_task, merge_input_values)

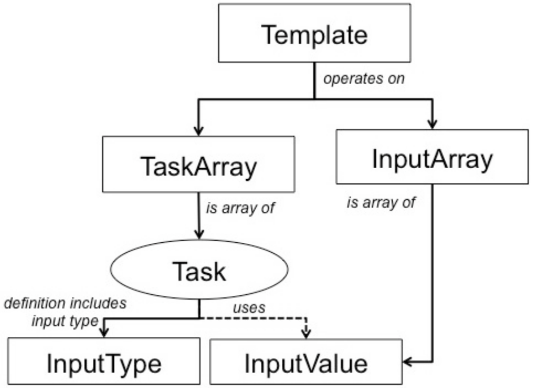
Tigres: Research Scope

- Programming interface to support workflows
- Optimize execution semantics on HPC systems
- Provenance and monitoring at scale
- Usability processes for API design and development

Tigres provides a “library” to support the iterative workflow development



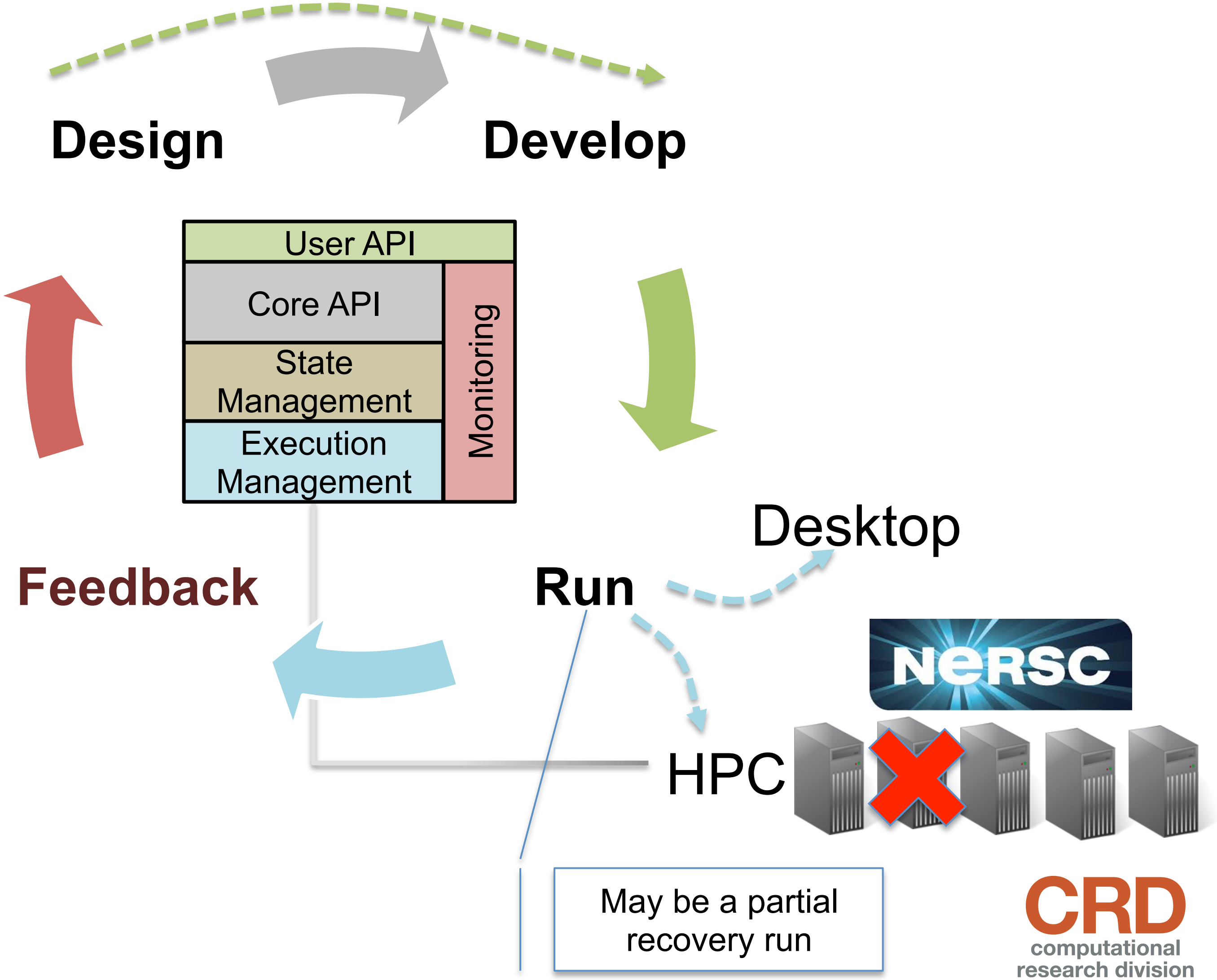




Tigres data model

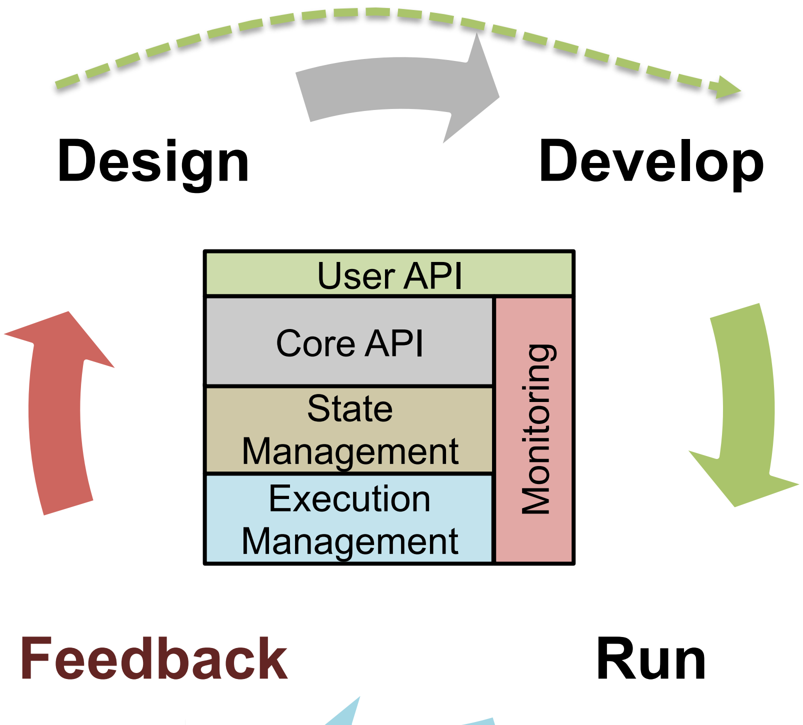
```
...
split(name=“Split”...)
merge(name=“Merge”...)
...
end()
```

Tigres provides a "library" to support the iterative workflow development



May be a partial recovery run

Tigres provides a "library" to support the iterative workflow development



Design

Develop

User API

Core API

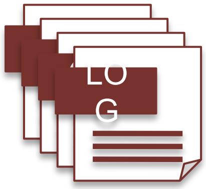
State Management

Execution Management

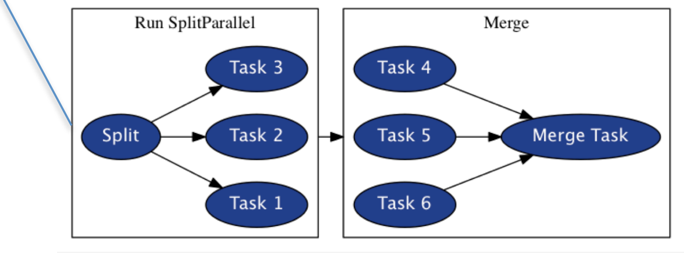
Monitoring

Feedback

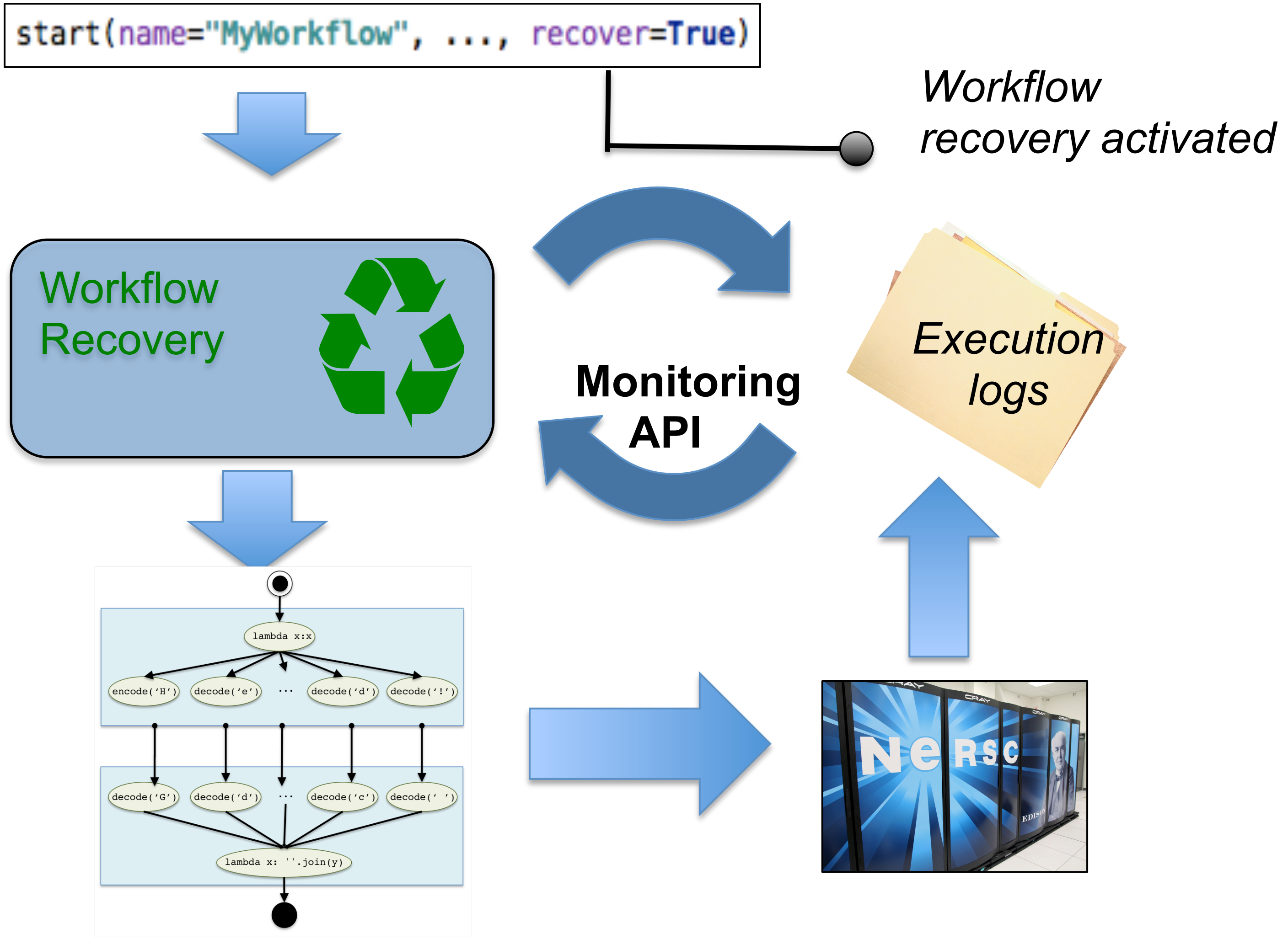
Run



Program state available during and after runs



Failure Recovery from logs



```
start(name="MyWorkflow", ..., recover=True)
```

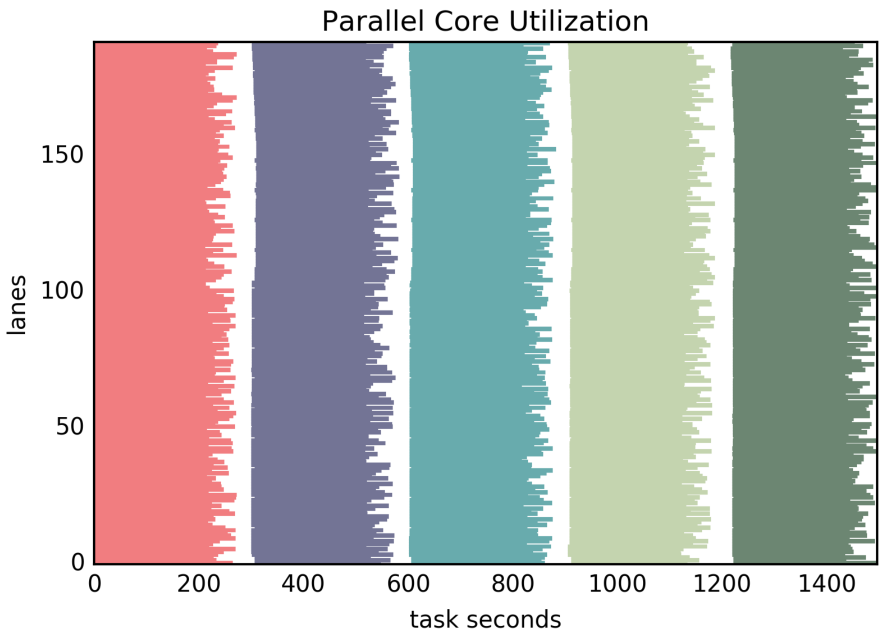
Workflow recovery activated

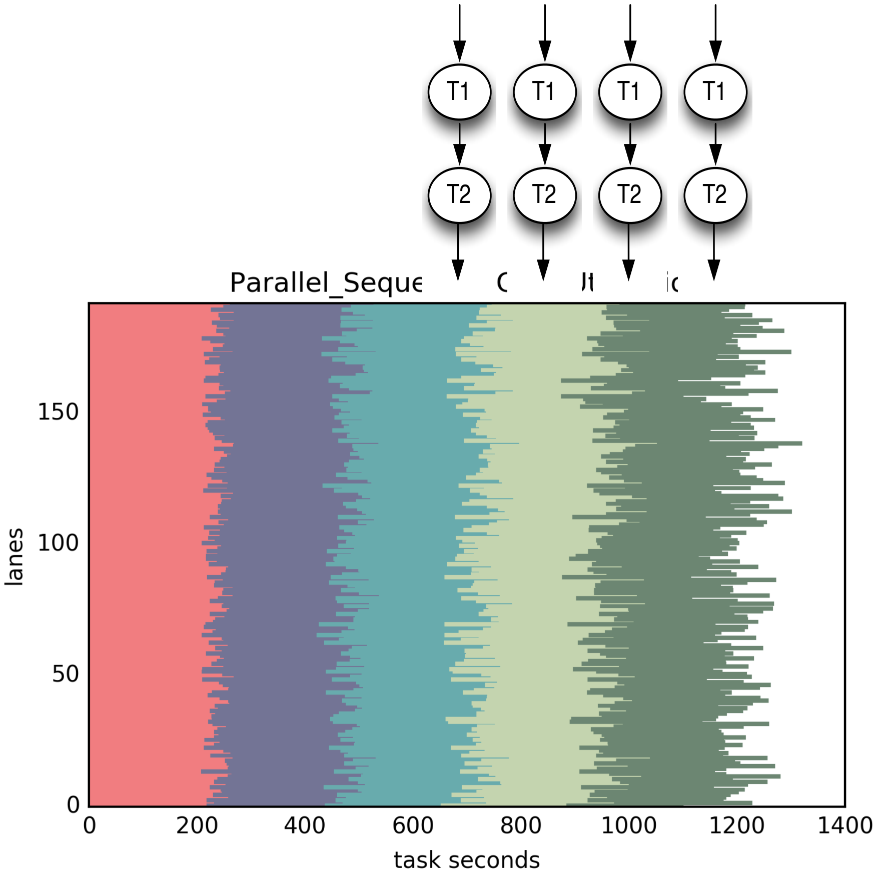
Workflow Recovery

Monitoring API

Execution logs

Parallel Sequential Performance Improvement

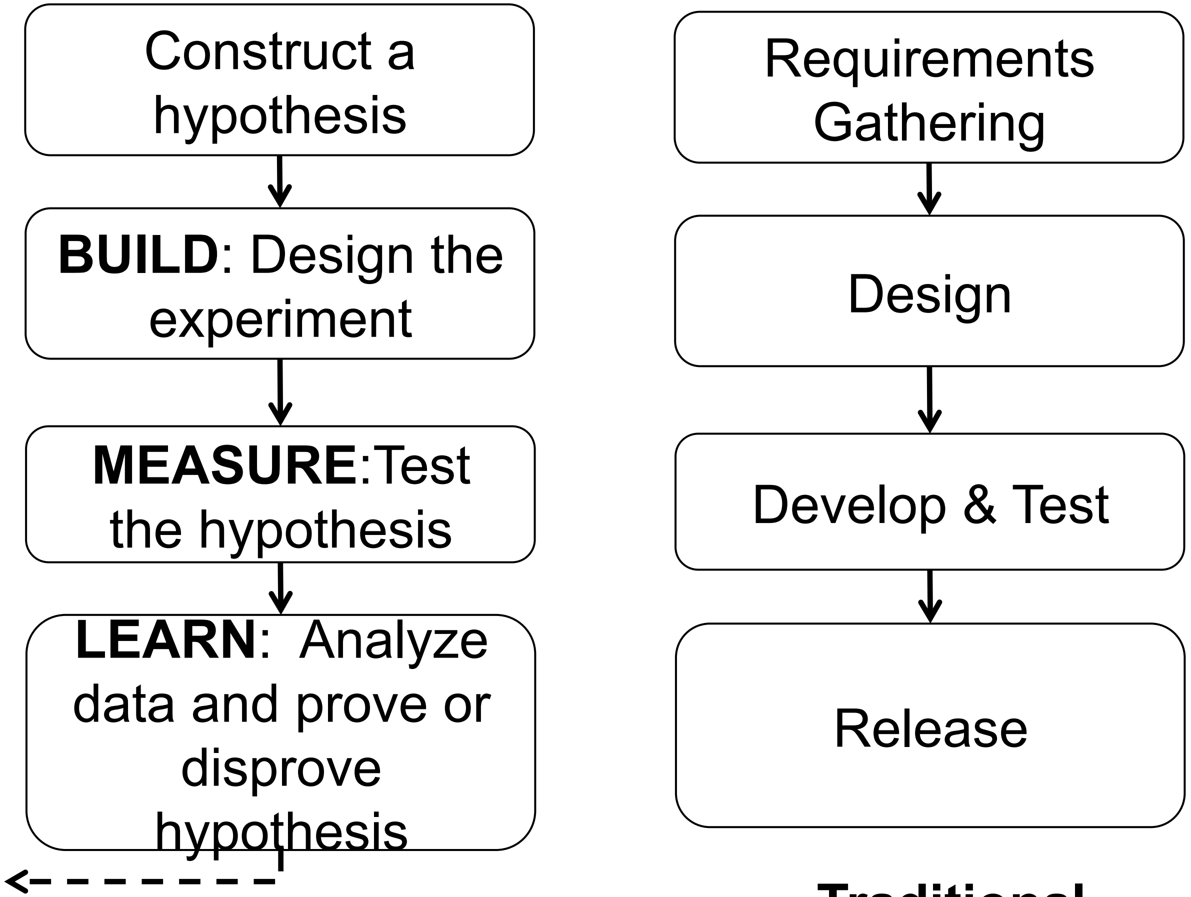




Template Time: ~11%

Resource Usage/Wastage: ~65%

Learning about the user as part of our process



Construct a hypothesis

BUILD: Design the experiment

MEASURE:Test the hypothesis

LEARN: Analyze data and prove or disprove hypothesis

Requirements Gathering

Design

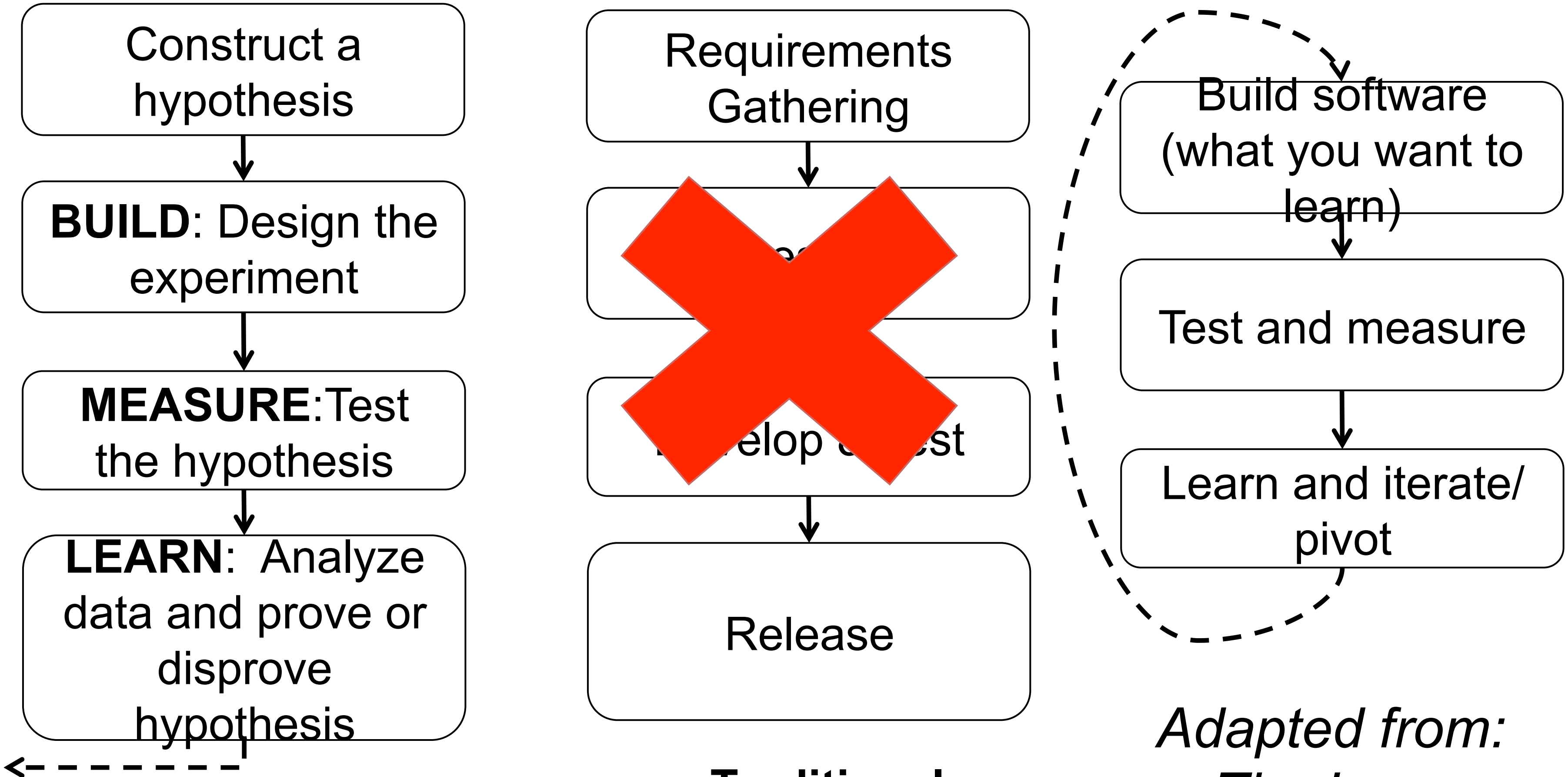
Develop & Test

Release

Research

Traditional Software Development

Learning about the user as part of our R&D



Construct a hypothesis

BUILD: Design the experiment

MEASURE:Test the hypothesis

LEARN: Analyze data and prove or disprove hypothesis

Research

Requirements Gathering

[illegible]

Develop & Test

Release

Traditional Software Development

Build software (what you want to learn)

Test and measure

Learn and iterate/ pivot

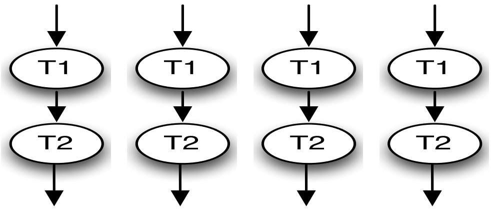
Adapted from: The Lean Startup Method

User-Centered Design Process [1/2]

- Usability studies provides semi-structured feedback from end-users
  - *Not the same as requirements gathering*
  - Limited literature on doing usability for APIs
- Round 1: Paper API & Google Docs Coding Session
  - Goal: Nomenclature and desired features
  - Topics from study: Concept understanding by user, Changes to Nomenclature, Support in C also important, Priorities for first prototype, Desktop to NERSC, Monitoring, Intermediate state management
  - Priorities: Nomenclature, Monitoring, Dependency syntax, ..

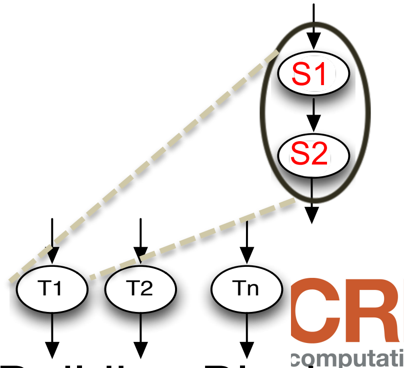
User-Centered Design Process [2/2]

- **Round 2a: Online Questionnaire after trying out Tigres**
  - + **67% said it was good and close to what they expected, 33% said it is definitely useful but needed to try it out**
  - + **20% thought it required more code than what they expected**
  - – **80% said minor learning difficulties**
  - – **40% said they would like more control**



ParallelSequential

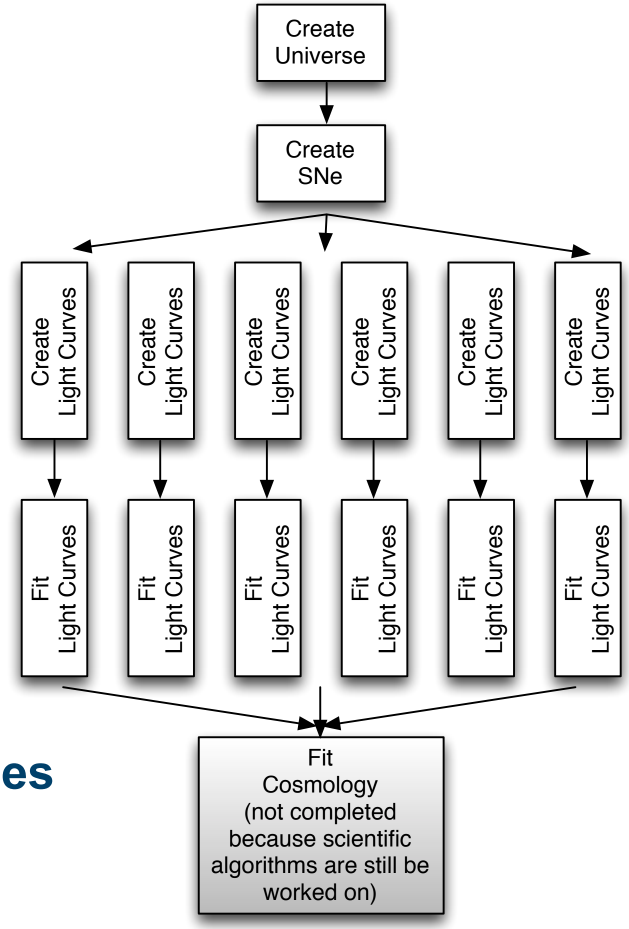
- **Round 2b: Interview and Post-task walkthrough**
  - **Support for nested templates**
  - **Investigation of running loops in Tigres**
  - **Difficulties with PREVIOUS syntax (including missing documentation)**



Building Blocks

Extensive Evaluation using Scientific and Synthetic Workflows

- **BLAST**
  - Bioinformatics workflow
  - One Parallel and two Sequences
  - 120 to 1800 tasks, Python executable
- **CAMP**
  - Satellite image re-projection
  - Two parallel and one sequence
  - ~6000 tasks, Python executable
- **Montage**
  - Astronomical Image Mosaic Engine
  - Three Parallel templates and two Sequences
  - C executables
- **SNe Simulation**
  - Cosmology
  - Python executable and functions



SNe workflow

U.S. DEPARTMENT OF ENERGY | Office of Science

CRD computational research division

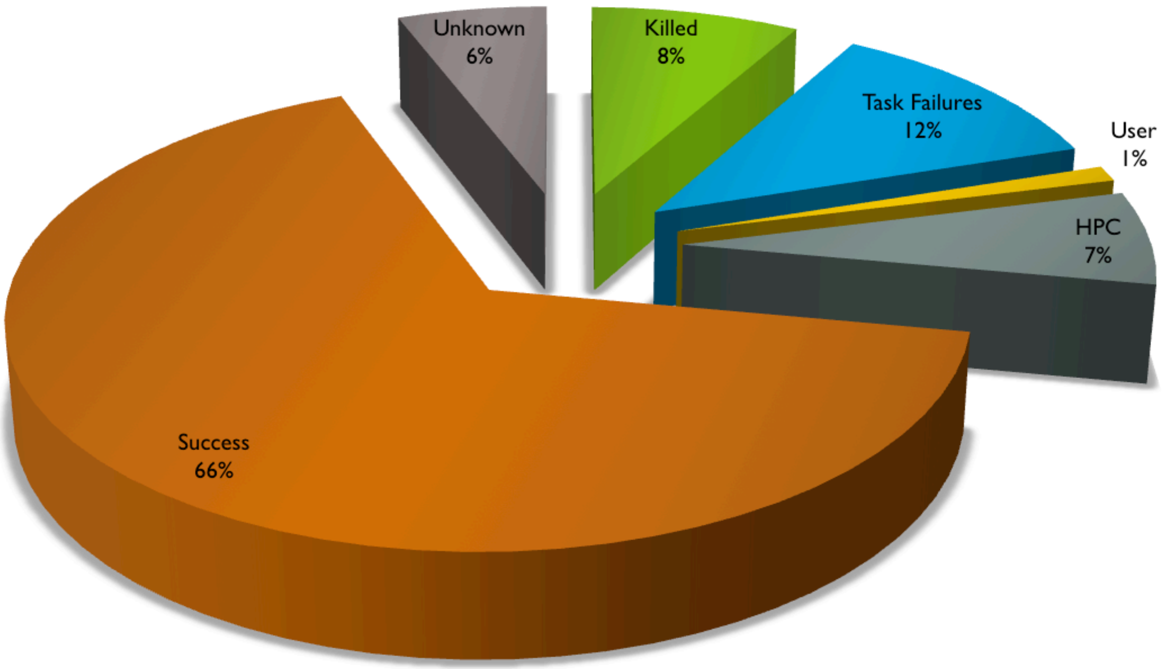
Experiences

- Setup of workflows is still tedious
  - libraries, diversity of resources
- Is portability from desktop to HPC achievable?
  - Code is not always developed for HPC
  - Queues policies and file system etc need to be understood
  - Understand characteristics for performance optimization
- Achieving efficiency is not trivial
  - Need to account for performance variability
  - Python setup performance (now improved at NERSC)
  - Different file systems' performance needs to be considered
  - [How our allocation "BLAST"-ed out]

Summary of Workflow Status [@ Tigres Level]

Workflow Status	Count	
Interrupted (Task failures in log)	18	1%
Interrupted (No failures recorded)	81	4%
Never started	169	9%
Failed (finished with failed state)	139	7%
Success	1575	79%
Total	**1982**	

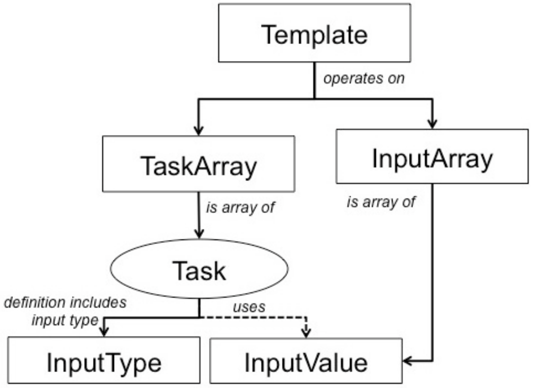
Summary of Job Status [@ Job Script Level]



Some Jobs have more than one workflow. 1982 workflows were submitted in 1160 jobs

Main Error	Error Detail	HPC Jobs	% HPC Jobs
Killed Job	Job-level	95	8%
	Terminated (Something or someone)	23	2%
	Wall-time Exceeded	72	6%
Task Failure	Workflow-level	137	12%
	Missing Files	104	9%
	Error Opening File	6	1%
	HPC Config	5	1%
	FTP Error	10	1%
	Other Errors	12	1%
User Error	Both levels	13	1%
HPC Error	Job-level	78	7%
	Broken Pipe	6	<1%
	IO Error	4	<1%
	Other	16	1%
	caught signal terminate	52	5%
Unknown	No error file/output	68	6%

Tigres: Feature Set



Tigres data model

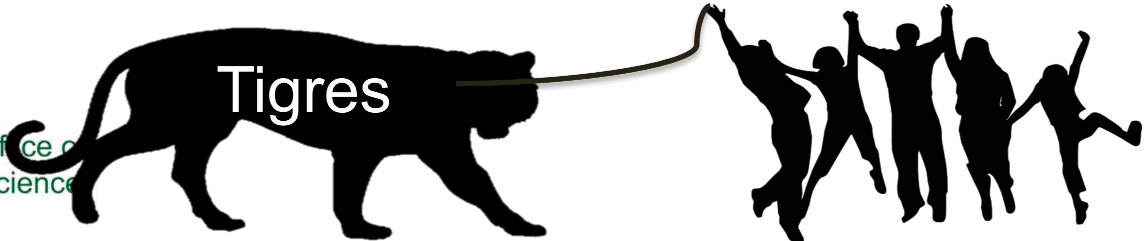
- Iterative workflow development
  - Simple data model
  - Python API to compose and execute
  - Use programming language constructs for complex logic flows
- Execution
  - Existing application binaries, functions
  - Seamlessly run on Desktops, Clusters and HPC
- Monitoring, Provenance
  - Visual representation of graph that ran
  - Extensive monitoring from workflow execution
  - Support for adding user-level provenance
- Extensive documentation, examples and tutorials
- Recover failed workflows from logs (Limited)
- C API (Limited)

Some Research Topics

- Active Code Generation
- Intelligent and Improved workflow management
  - Can we pipe the intermediate data ? (
  - Python backend is not optimal
  - C++/MPI could help in some cases and not others
- Deployment Configuration: Tigres + Shifter
- Better failure detection and reporting
- Synergistic
  - Workflow Scheduler at the Batch Queue Level
  - Managing data space for science workflows
  - Managing elastic environments for science workflows

Use of Tigres

- CAMP – Re-projecting MODIS data for 2010-2014
- TAKO - image processing software, SNe simulation group
- ARES/BDC – analyses pipelines for processing background radiation data
- Earth system simulation
- Inria Associated Team (frontend for HOCL)



Lessons Learned: Template Interface

- Python interface was very attractive for many of our early users
- Template interface was also attractive for simple DAGs
  - Is there a specific way I should split my workflow into templates?
  - Very few cases where they had unusual DAGs
- Nested templates was a key feature request
  - ParallelSequential was a good example
  - General nested template needs more
- Template/Interpreted language – no global view of DAG and other programmatic modifications to data.

Lessons Learned: Straddling the Research and Software Development Boundary

+ User-Centered design process enabled us to receive valuable “early” feedback

+ The user-centered design process forced us to address S/W development lifecycle in a research project early

? Users wanted access to software which presents challenges in a research project.

? Need to reduce the time in the cycle of build, measure, learn and balance the cycles of learning about the user and CS research

Looking forward …

- Tigres provides a good foundational tool for many users and experiments
- Developing and communicating best practices
  - User-centered approaches for software/middleware development
  - Lot of what we have learned are lessons for users outside of workflow tool (e.g., Python is not suited for all tasks)
- Near-term research
  - How are we going to support programming “data” workflows?
  - Human-in-the loop issues
- Long-term: “workflow tools” need to disappear
  - More support at infrastructure level and application programming models?

More Information

- This work is supported by the DOE Office of Science (Office of Advanced Scientific Computing Research)

- Tigres Team
  - Lavanya Ramakrishnan, Valerie Hendrix, Sarah Poon, James Fox, Gary Kushner
  - Ryan Rodriguez, Daniel Gunter, Gilberto Pastorello, Deb Agarwal

- Lramakrishnan@lbl.gov

- http://tigres.lbl.gov

Tigres C

- Current Implementation
  - C API with a Python backend
  - Macros used to define functions
  - The fully expressivity of PREVIOUS is not implemented
- Food for thought
  - Performance of Python
  - Parallelization of functions and Deserialization of data
  - C does not posses a runtime type introspection (Do you manage to keep consistency with Python?)
  - Usability - “Pythonic”/C-like code